A MODUS OPERANDI
WHITE PAPER

Modus Operandi

Applying Wave® to Build an Air Force Community of Interest Shared Space

by Karen Dyson
Wave Product Manager

122 Fourth Avenue
Indialantic, FL 32903
USA
321-984-3370
www.modusoperandi.com

August 2007

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE **01 AUG 2007**	2. REPORT TYPE **N/A**	3. DATES COVERED **-**
4. TITLE AND SUBTITLE **Applying Wave to Build an Air Force Community of Interest Shared Space**		5a. CONTRACT NUMBER
		5b. GRANT NUMBER
		5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)		5d. PROJECT NUMBER
		5e. TASK NUMBER
		5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Modus Operandi, Inc 122 Fourth Avenue Indialantic, FL 32903 USA**		8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)		10. SPONSOR/MONITOR'S ACRONYM(S)
		11. SPONSOR/MONITOR'S REPORT NUMBER(S)
12. DISTRIBUTION/AVAILABILITY STATEMENT **Approved for public release, distribution unlimited**		
13. SUPPLEMENTARY NOTES		
14. ABSTRACT		
15. SUBJECT TERMS		

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**	**SAR**	**20**	

Copyright

Restricted Rights Legend

Trademark or Service Marks

Acknowledgement of Support

This material is based upon work supported by the United States Air Force under Contract No. FA8750-04-C-0082. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the United States Air Force.

This material is based upon work supported by the Air Force Research Laboratory under Contract No. FA8750-04-C-0068. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the Air Force Research Laboratory.

Table of Contents

Abstract

This paper describes the application of Modus Operandi's Wave product to the construction of a Community of Interest (COI) shared space. An example of this is the Knowledge Management Framework (KMF) at the 45th Space Wing (45 SW) of the US Air Force Space Command at Cape Canaveral Air Force Station and Patrick Air Force Base. The KMF is a shared space for the Eastern Range community of interest.

Wave addresses key elements of a COI shared space, including a registry, catalog, virtual data storage mechanism, and vocabulary deployment within a bounded network space. A shared space can serve as reusable infrastructure for a variety of services and applications that serve the COI.

This paper attempts to answer the following questions:

- What steps are required to build a COI vocabulary for a shared space?
- How can that vocabulary be made operational to support a shared space?
- What are the benefits?

Throughout this paper, a number of key best practices are identified as part of an overall methodology for the efficient, effective, and repeatable application of Wave to the task of building a COI shared space. The Wave product and supporting methodology have a number of unique facets, but a primary differentiator is this:

Wave makes it possible to operationalize a COI vocabulary that is machine-processable and dynamically populated at runtime to serve as a shared space.

Introduction

A Roadmap of this Paper: Establishing Context

This paper covers two primary areas: the construction of a COI vocabulary for a shared space service oriented architecture (SOA) data services layer, and then how that vocabulary can be made operational to serve the COI's stakeholders as a shared space. The construction, operation, maintenance, and extension of a variety of end-user applications, which consume information from this common infrastructure or shared space, are being developed independently and are not explored here.

Community of Interest

In this paper, Modus Operandi uses terminology consistent with Department of Defense (DOD) Communities of Interest. DOD 8320.2, Data Sharing in a Net Centric Department of Defense, provides the following definition of a Community of Interest:

A collaborative group of users that must exchange information in pursuit of its shared goals, interests, missions, or business processes and therefore must have shared vocabulary for the information it exchanges [1].

Shared Space for an Air Force COI

A shared space for a COI is a mechanism that provides storage for and access to data for users within a bounded network space. A shared space provides virtual or physical access to any number of information assets (e.g., catalogs, Web sites, registries, document storage, and databases). For example, using a library analogy, the bookshelves in a library, or the library itself, are a shared space. A virtual library may be manifested as a repository that contains copies of, or links to, the actual holdings in the library. Registry content and catalog content are held in a shared space [2].

In the case of the work done with the 45 SW Knowledge Management Initiative (KMI), the 45 SW was not a chartered Air Force (AF) COI at the inception of the project. Work was performed by the 45 SW KMI team, including AF personnel, support contractors, and Modus Operandi. Modus Operandi led the infrastructure design and construction, the vocabulary construction (modeling), and building of applications to consume information assets in the Knowledge Management Framework shared space.

Building a Vocabulary

In this chapter, we focus upon methodologies and best practices for developing a vocabulary for a COI. This effort must be driven by a well articulated business case in order to succeed, both in terms of technical feasibility and potential for adoption. Given a solid business case, five key decisions in the vocabulary development process are presented.

The Business Case

The construction of a vocabulary must be driven by a specific business problem and related competence questions. This focus is essential to a successful outcome.

The Business Problem. One example of a mission objective is to answer the following question:

Is the Eastern Range fully mission capable for missions scheduled in the next 90 days?

Existing business processes that answer this question may take too long and be too labor-intensive. As a consequence, commanders must rely upon status information with a staleness of 12-24 hours. For instance, instead of a daily situation report (SITREP), a key improvement would be up-to-date status visible at any time. Thus the business problem is the need for automation of an existing business process to improve the timeliness of the information.

Understanding the Question. The “Eastern Range” phrase in the question above relates to range instrumentation status, and is distinct from launch readiness. Range instrumentation resources are owned and managed by the range. Launch readiness depends on additional factors such as launch vehicle and payload status—resources owned and managed by the launch customers. The phrase “mission capable” is always answered in the context of a particular mission, because it would be possible to be fully mission capable for one mission while at the same time not mission capable for another mission. This is possible because different missions require different sets of range instrumentation resources.

Competence Questions. The goal is construction of a vocabulary, linked to Authoritative Data Sources (ADS’s) in order to answer the following questions:

Q1: What missions are scheduled within the status window?

Q2: What resources are required for a particular mission?

Q3: What is the status of those resources?

Understanding the Processes. To find ADS’s to support the resolution of each competence question, one must start by understanding the business processes and tracing the threads of information flow. This is best done by interviewing subject matter experts within the COI and examining existing business process work products.

Selecting the Authoritative Data Source. When seeking to select ADS's, it is not enough to examine a schema or an Entity-Relationship Diagram (ERD), or to read the user documentation. It is essential to interview system users to understand how they use the system in day-to-day operations.

Some of the questions that should be asked regarding candidate ADS's include the following:

- What fields do they routinely ignore?
- What fields did they adapt for a purpose other than what was originally intended?
- What features do they not trust?
- How do they work around missing capabilities?
- Have they changed the way they use the system over time?

The developers own the schema, but the users own the meaning!

Five Major Decisions

Five major decisions comprise this effort:

- What information assets to manage and expose
- How to model semantics
- When to stop modeling
- How to access the authoritative data sources
- How to handle semantic conflicts

Decision 1: What Information Assets to Manage and Expose

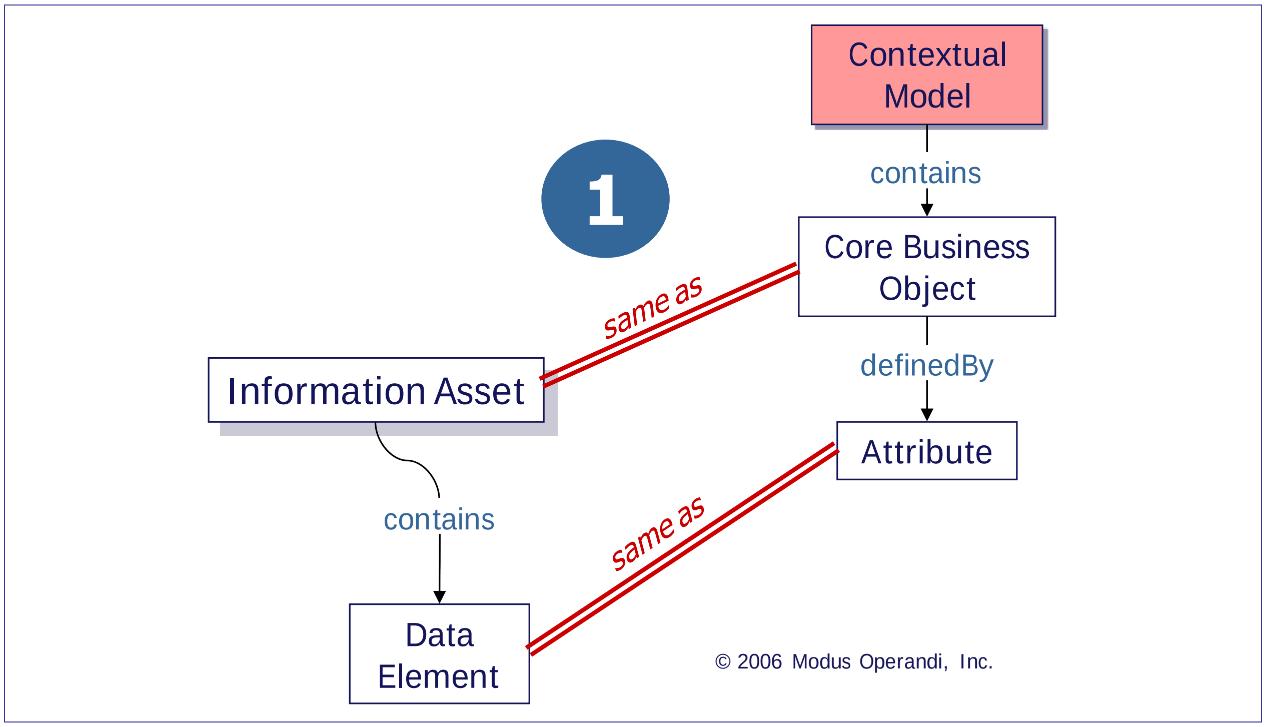

Diagram 1 – Where to Start. Determining what information assets to manage and expose is a pivotal decision.

This is a pivotal decision, from which other decisions will flow. Quality and availability of existing data sources play a major role, as does the state of automation of existing business processes. In the case of the KMF at the 45 SW, we selected core business objects as the information assets to manage and expose, as opposed to documents. Core business objects include concepts such as mission, instrumentation system, facility, and operation.

Where business processes do not have the benefit of automated support, information may not be exchanged in electronic form, but instead via a phone call or sticky note, or by distributing hardcopy. Where documents are exchanged, they might be email messages, or Microsoft Office products such as documents, spreadsheets or presentations. Trying to extricate the information content from these sources is a significant challenge, targeted by everything from Extensible Markup Language (XML) to text analytics and entity extraction tools, each with varying degrees of success.

Decision 2: How to Model Semantics

Combining the Models. We have found significant value to an approach which *combines* the semantic and contextual models as shown in Diagram 2 and expresses them in Web Ontology Language (OWL). Reasons include the following:

- Desire for a vendor-neutral, machine-processable representation that can be used to generate data services at design time as well as to be consumed at runtime. (For more about this, see Generating Services at Design Time)
- Frequent requirements to link the two models explicitly.
- Difficulty drawing a clean boundary between the models, as the combined model is used to resolve semantic conflicts. (For more about this, see Decision 5: How to Handle Semantic Conflicts)
- Frequent requirements for a "live" or dynamic taxonomy that can be pulled from authoritative data sources. (See Settling Taxonomy Debates below)
- Potential to add more expressive semantics later, as needed.
- Anticipation of the value of reasoning over information in the shared space.

Settling Taxonomy Debates. There may be debates between the taxonomy purists (who insist that a taxonomy be a strict generalization/specialization or "isA" hierarchy) and the pragmatists who observe the subject matter experts categorizing objects in their domain in a mixture of ways. In practice, we recommend concentrating on modeling *what is in use* rather than *what we would like it to be*, and thus the following guideline:

When you find an authoritative data source for part of a taxonomy, stop modeling. Start integrating!

Live Taxonomy. Instead of documenting a taxonomy in a spreadsheet, semantic infrastructure requires a machine-processable format. Instead of populating a static OWL model with all the elements of a taxonomy down to the leaf nodes, we recommend integration to ADS's to populate elements of the taxonomy dynamically at runtime. For example, instead of enumerating the types of instrumentation systems in a static model, one would extract the list of types from the system maintenance data source. This dynamism eliminates worry over staleness or data entry transcription errors. In addition to the inherent benefits of dynamism, this approach has the virtue of settling some of the modeling debates, which leads to the next decision.

Decision 3: When to Stop Modeling

A model is a representation of something for the purpose of explaining or predicting its behavior. To know when to stop modeling, the modeler needs a way to judge what fidelity the model must have to the thing it represents in order to be useful. An architect builds a scale model to impress his clients or to help those who can't visualize three dimensions from a blueprint. A structural engineer builds a finite element model to design a bridge, and refines the mesh until the results converge. It helps to have heuristics to determine when the model is good enough.

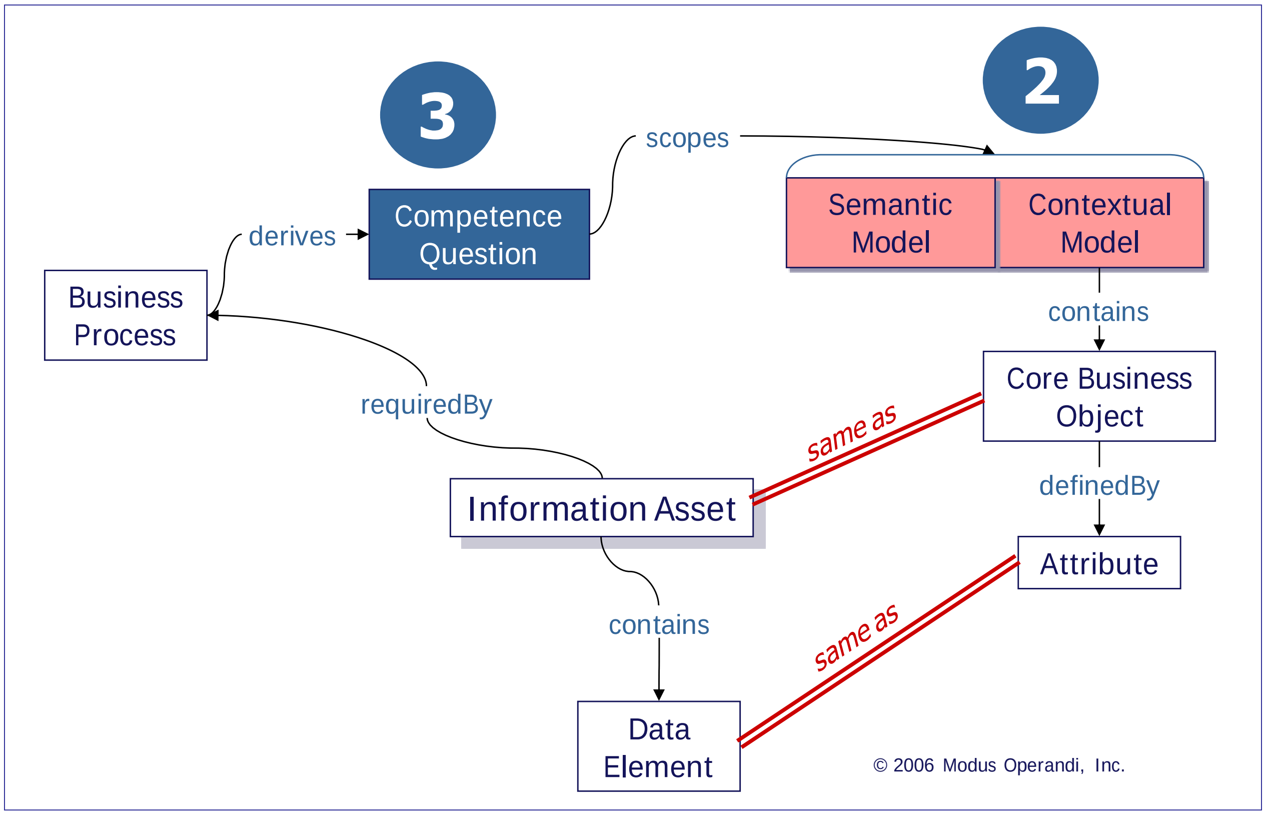

Diagram 2 – "Good Enough" Model. Focusing on a business problem allows better scoping of the modeling activity and determination of when the task is finished.

The scope of the model can be expressed by the following dimensions:

- Breadth of model—included business objects
- Depth of model—relationships linking the business objects, and attributes describing them
- Semantic richness—taxonomic and composition structures

The model has two related purposes that drive the scope and together define what is good enough.

- Solve a business problem
- Resolve semantic conflicts among authoritative data sources required to solve the business problem

Competence questions are used to characterize what information the model needs to provide at runtime in order to solve the target business problem. We strongly recommend against a big bang modeling activity, or a one-problem-one-model solution. In practice, it is most efficient to solve a number of business problems and model iteratively.

The key to success is not to try to model
the whole world up front.

Decision 4: How to Access the Authoritative Data Sources

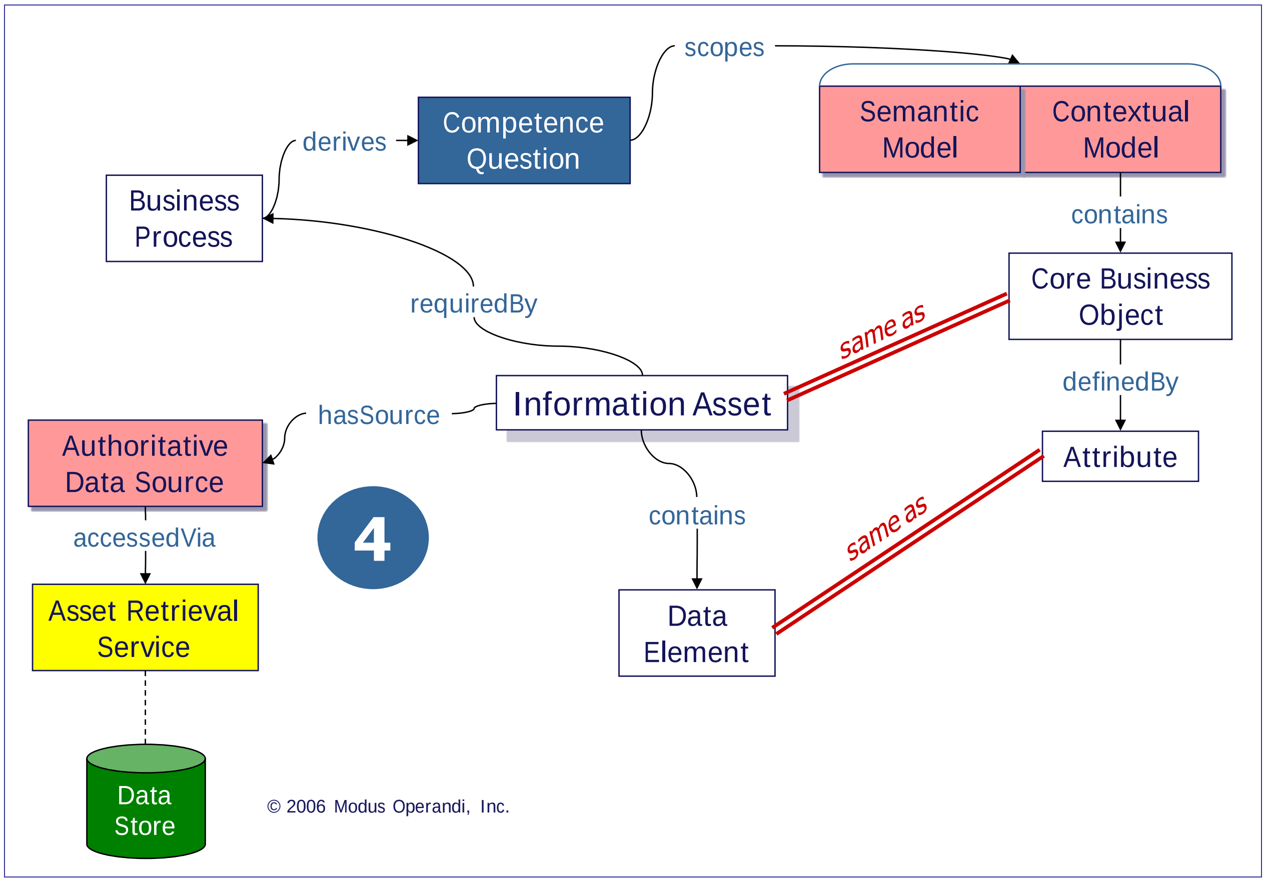

Diagram 3 – Quest for the ADS. It can be very difficult to identify truly authoritative data sources when business processes are not very automated. When you do find them, they are likely to be stovepiped systems.

Not a Simple Mapping. A characteristic of Stovepipe City is that information about a business object may live in more than one place. For example, some attributes may come from one ADS while others come from another ADS. There may be duplication or outright disagreement. Therefore one cannot rely on a simple mapping from a type of information asset to an ADS. Part of the job of identifying ADS's entails dealing with these challenging issues.

Ownership and Conflict. Just because two ADS's are owned and maintained by the same organization doesn't mean that they will be conflict-free. In most virtual enterprises or COIs whose membership crosses organizational boundaries, ADS's may have different owners. Where there is overlap, it may be more valuable to focus on bi-directional mediation, rather than allowing the conflict to escalate and then declaring a victor.

Decision 5: How to Handle Semantic Conflicts

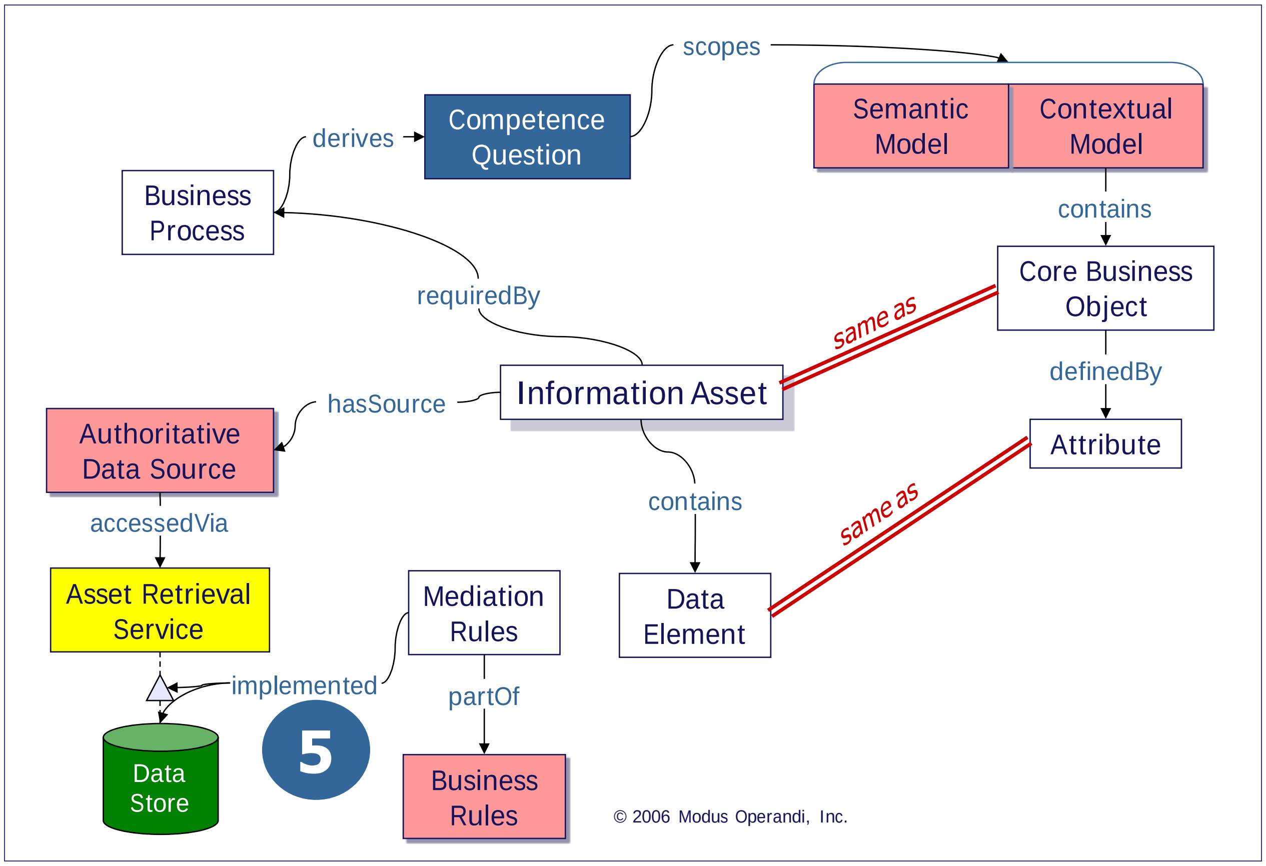

Diagram 4 – ADS's might not work together without help. Metadata alone does not bridge stovepipes. Mediation is necessary to resolve semantic conflicts among authoritative data sources within a COI.

Stovepipe City. One can expect that there will be conflicts among the different stove-piped systems in their representations of core business objects. In fact, one of the characteristics of living in a Stovepipe City is that each stovepipe acts like it is the only authoritative data source! (Stovepipe City is not the name of an actual town; rather, it is all around us.)

Model Mapping. We recommend a model-mapping approach, mapping from an ADS to a canonical model in a logical hub-and-spoke pattern. In this scenario, the Contextual Model is the canonical model hub. Why use this pattern? A model-mapping pattern is one of the most versatile and practical ways to overcome a wide range of semantic conflict types. For more information about this topic, Pollock provides an analysis of the types of semantic conflict and their resolution techniques in chapter 5 *Adaptive Information* [3].

In practice, one may use a variety of techniques for resolving semantic conflict, and many are easy to apply with the design time tools. The following examples, however, aren't as simple, and must be implemented with care so as not to degrade query performance.

- Custom transformation function (such as Java code) – for example, to convert a non-standard date field to a standard date representation.
- Lookup table or link table – for example to mediate unique identifiers among ADS's.

Resolution can be Costly. Resolving semantic conflicts is one of the major costs of "on-boarding", i.e., integrating, a new ADS into the shared space. A benefit of the model-mapping pattern is that you only have to do it once, and then all future consumers of the ADS information can reuse it.

Resolving each semantic conflict is like winning a battle.
Achieving semantic integration is a big victory
for the COI.

Mediation. In the context of this paper, mediation is achieved through mapping and transformation to and from the canonical model. It was beneficial to think about and do both mapping *and* transformation at the same time in the same tool environment. (See also Generating Services at Design Time.)

Mediation and Query Performance. It is essential that an inverse transform be defined for every transform, or else the query mediator must be smart enough to figure out how to invert it. Without an inverse transform, if an incoming query constrains on the transformed attribute, the query mediator might generate a query plan that is horribly inefficient. If you must code a custom transformation function, you must also code the inverse transform.

Putting the Vocabulary to Use

The driving business case and five major decisions presented in this chapter provided the necessary context and guidance to build the COI vocabulary. The next chapter illustrates how the vocabulary comes alive in the shared space infrastructure.

Building a Shared Space

This chapter focuses upon how we make the COI vocabulary come alive. The following topics are covered:

- Off-the-shelf infrastructure components
- Generating services at design time
- Out-of-the-box runtime services
- Keyword search support
- Scaling considerations

Off-the-Shelf Infrastructure Components

The following components support the shared space infrastructure:

- Modus Operandi Wave®
  - Design Time Components
    - Wave Importer for WebLogic® Workshop – generates data services from OWL
  - Runtime Components
    - Wave Web Service Application Programming Interface (API)
    - Wave Indexer
    - Wave Search – a keyword search web application
    - Wave View – a visual model navigation and instance browsing applet
    - Wave Console – a web app for developers to browse the data services' structural metadata, monitor the status of the data services, and run test XQueries
    - Wave Watch – a health and status monitor for all data sources
- BEA AquaLogic® Data Services Platform (DSP)
- BEA WebLogic Platform®
- Protégé – for ontology construction

Generating Services at Design Time

Core data services are generated from the COI vocabulary at design time. Diagram 5 below shows what elements of the vocabulary were used to generate the core data services.

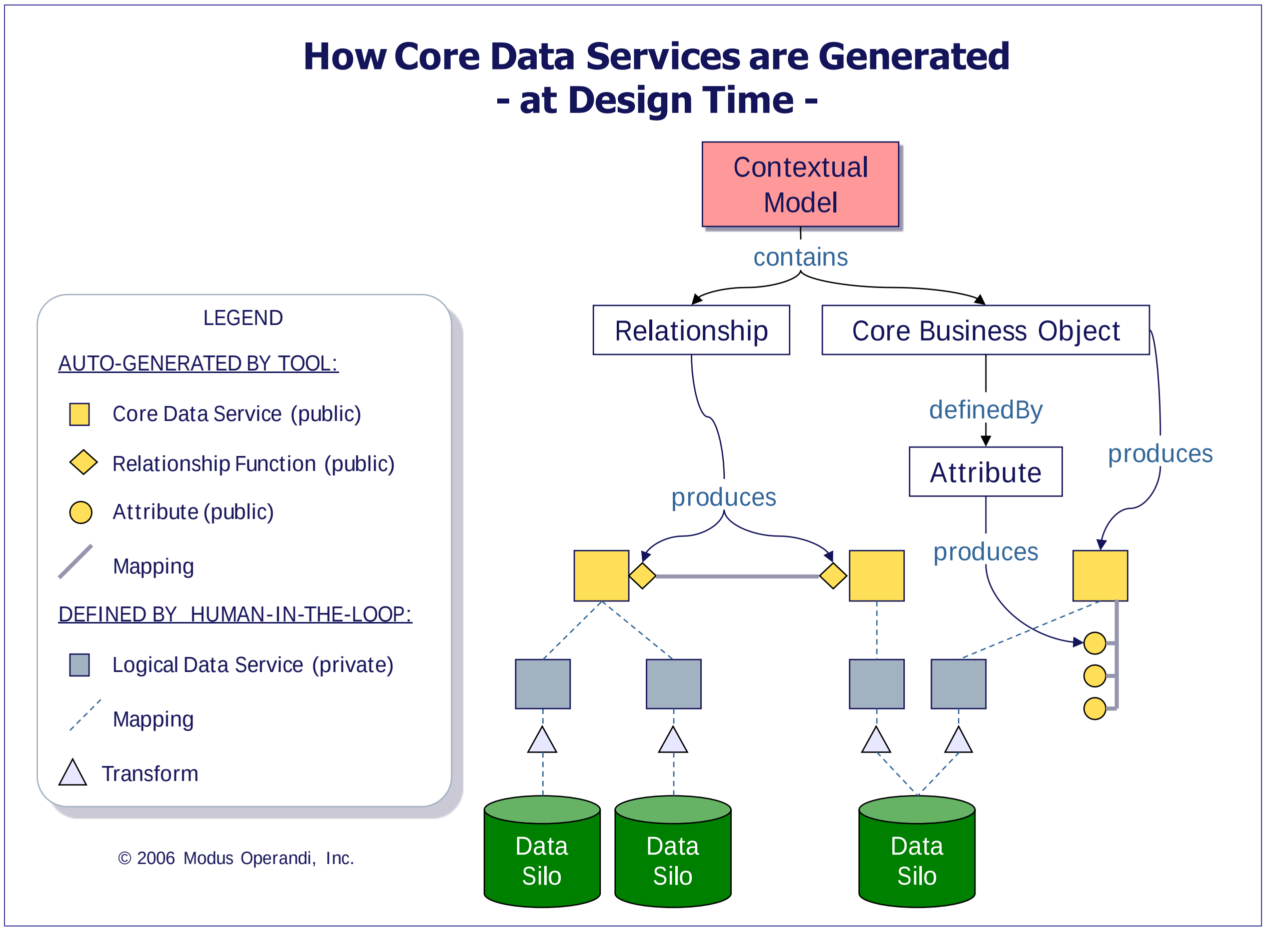

Diagram 5 – No Silver Bullet. Tools greatly enhance design time productivity, but do not fully automate the process.

Data Access Service Generation. Modus Operandi's Wave Importer for BEA WebLogic Workshop consumes an OWL model to generate core data services. As depicted above, each business object class produces a logical data service. The attributes of the class produce the attributes of the service. A bi-directional relationship produces two relationship functions, one for each class. On the left side of the diagram, the legend indicates which parts are automatically generated by Wave, and which parts are created or mapped by a user.

The design time tools help the user with mapping and transformation. But they do not automatically link directly to the data sources. Why not? Because there may be semantic conflicts to resolve, and it is up to the user to identify and resolve them. In this way, the user identifies and implements the mediation rules. We recommend mapping to the core logical data services in a layered approach as illustrated and discussed in the next section. The layering helps to separate the hub from the spokes when applying a model-mapping pattern.

If there were no semantic conflicts to worry about (i.e., the ADS's were a perfect match to the canonical model), a tool could do the linkages automatically, however this would amount to simple schema introspection. Unfortunately, this "perfect world" scenario is seldom, if ever, the case.

The Role of Semantics. The semantic elements of the model support the following capabilities:

- Composition structures. If these are modeled as relationships between business object classes, then they produce relationship functions.
- Taxonomy.
  - If a parent class has attributes, then those attributes produce attributes in all child classes. This provides a mechanism to enable queries against an abstract parent class. The user must map the child class attributes to the parent class.
  - To simplify implementation, Wave does not propagate parent class relationships. Otherwise, this could easily become an unwieldy mass of relationship functions. If the parent class has M relationships to other classes, and has N child classes, then there would be N x M functions generated.
- Ontology. The OWL ontology is saved and is available for discovery.

Out-of-the-Box Runtime Services

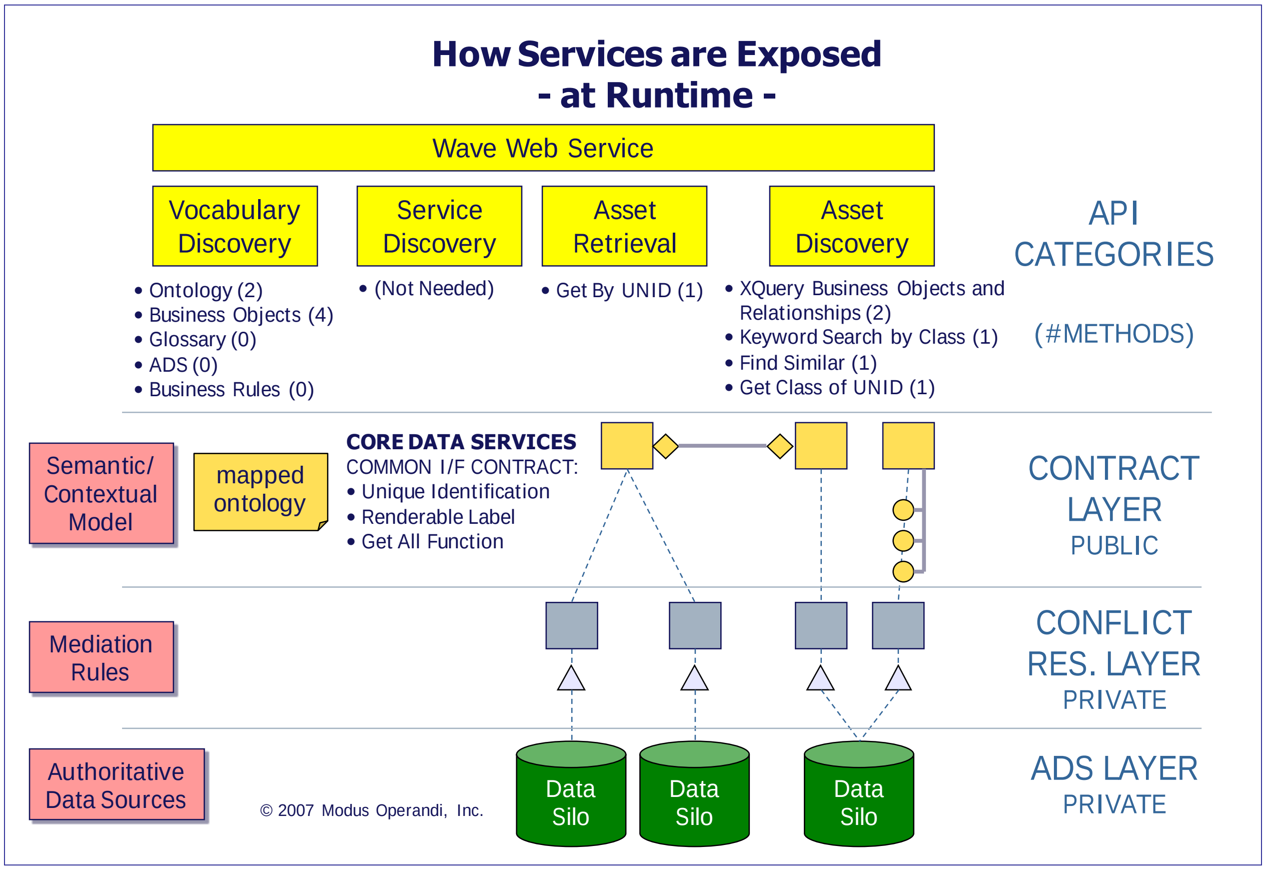

Diagram 6 – A Virtual Metadata Catalog. The mapped ontology—an OWL file—serves as a metadata repository. Metadata about instances is retrieved as needed directly from data sources.

COI shared space services are exposed at runtime via the Wave web service API. The Wave API provides discovery and retrieval capabilities for the operationalized COI vocabulary and federated ADS's.

Wave Application Programming Interface Categories. The diagram above categorizes the API methods.

- Vocabulary Discovery. The Wave API provides methods for discover the ontology and related business objects.
- Service Discovery. A UDDI registry for the core data services is optional, or perhaps not necessary at all, because the services are linked to the business objects in the vocabulary. In essence, activating the COI model takes the place of a registry. In the future, the desire to share the implementation of higher tier services that consume the core data services will drive the need for a registry.
- Asset Retrieval. The only Wave API method shown here is Get by UNID (formally named `getObjectFromUNID`). Once you have a handle to the object, i.e., a reference to

an instance of a business object, you can retrieve all the attributes of the object. The Wave API's Asset Discovery methods provide handles to the objects.

- Asset Discovery. There are two basic ways to find objects:
  - XQuery. Because Wave exposes the shapes (XML Schema Definition (XSD)) of business objects in the contract layer, we can use XQuery to find objects by their attributes and to traverse relationships among objects.
  - Keyword Search. Wave includes a Google™-like API. All indexed attributes are searchable. Wave also has a Find Similar (`searchSimilar`) method that uses the values of the attributes of the selected object as search terms to find similar objects. More details about keyword search are provided in the next section.

Layered Approach. The public layer is exposed by the Wave API; the private layers are not.

- Contract Layer. A common interface contract for all of the logical (core) data services in this layer is required, and the necessary hooks are generated by the Wave Importer for WebLogic Workshop at design time. In addition to the core data services, this layer includes the complete OWL ontology.
- Conflict Resolution Layer. Mapping and transformation occurs in this layer to resolve semantic conflicts among ADS's, or between an ADS's unique representation and the idealized representation of the canonical model as expressed in the contract layer. The transformation is kept as close to the source as possible.
- ADS Layer. Connections to the ADS's live in this layer. This keeps the ADS's well isolated from the public interface, keeping consumers of the services from binding to the ADS's schema.

Mediation is the Second Choice. Performance and scalability concerns apply to any approach for an enterprise data layer. Mediation can often incur substantial performance and scalability overhead, thus one would generally prefer to avoid it. In most COI shared spaces, it is a necessity.

Mediation is the second choice. The first choice is to have an ADS produce a view that matches the canonical model, eliminating the need for runtime mediation.

If the preferred approach is feasible, then it should be pursued. If, on the other hand, your canonical model is subject to constant change, mediation may have more appeal.

Keyword Search

Keyword search is implemented by producing a semantically-enhanced index. The diagram below shows how the Wave index service crawls through the active model to generate an index. Wave's common interface contract for the core data services makes the business objects crawl-able and index-able, and ensures unique identification of the object instances. All of the attributes are indexed, including attributes mapped to a Character Large Object (CLOB).

The Wave search methods can be used to search all classes, or to constrain on a selected set of classes.

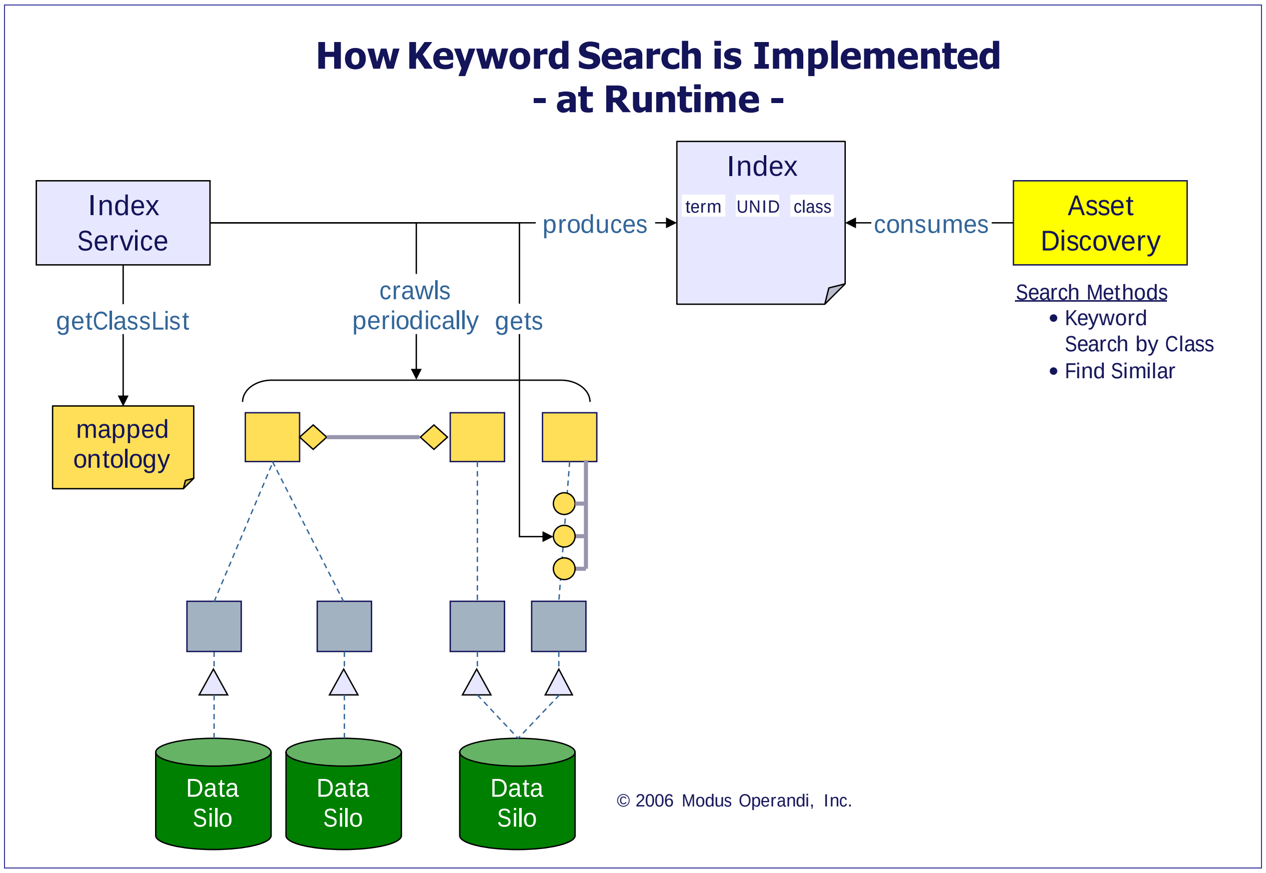

Diagram 7 - Indexing for Keyword Search. The Wave index service periodically crawls all classes, indexing the attribute values for every instance. Adding a class identifier to the index enables consumers of the asset discovery service to constrain on class, as well as to search all classes.

Scaling Considerations

There are many dimensions to consider when discussing scale. In terms of the number of simultaneous users, Wave relies upon BEA AquaLogic Data Service Platform linear query throughput performance to deliver scalability in that dimension. In terms of the size of the model, it is common to hit the human modeler’s cognitive ability limit long before the upper bound for the tools. The Wave semantic layer atop BEA AquaLogic Data Services Platform has been found to add minimal overhead to the existing baseline measurements identified by BEA. Additional details are available in the BEA white paper “BEA AquaLogic Data Services Platform Performance: A Benchmark-based Case Study White Paper” [4].

Conclusion

Benefits and Best Practices

Limiting the vocabulary to a machine-processable representation, equating information assets to business objects, and concentrating on structured data sources is essential to focus and scope the effort, to deliver value sooner, and to evolve the vocabulary incrementally. This approach was instrumental in the formulation of a repeatable methodology for the application of Wave, as presented in this paper.

Key Benefits

- Wave provides a way to operationalize a COI vocabulary, facilitating better leverage of COI assets and knowledge.
- Out of the box, Wave provides indexing, search, and navigation of all federated COI data. This yields a rapid return on investment (ROI) from the establishment of the COI shared space, even before specialized services and applications are built.
- A shared space can serve as reusable infrastructure for a variety of services and applications that serve the COI. This avoids the unnecessary repetition and maintenance headaches of typical, single-use integration efforts that serve only one application or tool.
- The Wave product and supporting methodology offer a reliable and repeatable approach to the establishment of a COI shared space.

Best Practices. This paper has elaborated upon a number of important best practices, which are restated here:

- The developers own the schema, but the users own the meaning!
- When you find an authoritative data source for part of a taxonomy, stop modeling. Start integrating!
- The key to success is not to try to model the whole world up front.
- Resolving each semantic conflict is like winning a battle. Achieving semantic integration is a big victory for the COI.
- Mediation is the second choice. The first choice is to have an ADS produce a view that matches the canonical model, eliminating the need for runtime mediation.

References

[1] DOD Directive 8320.2, Data Sharing in a Net Centric Department of Defense, December 2, 2004.

[2] “Air Force Metadata Environment Concept”, Working Draft, Version 1.0, November 8, 2006.

[3] Pollock, Jeffery and Ralph Hodgson, *Adaptive Information*, Wiley, 2004, ISBN 0-471-48854-2.

[4] “BEA AquaLogic Data Services Platform Performance: A Benchmark-based Case Study White Paper”, http://www.bea.com/content/news_events/white_papers/BEA_AL_Data_Services_Plat_Peformance_2_5_wp.pdf.